View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Electronic Communications of the EASST (European Association of Software Science and...

Electronic Communications of the EASST
Volume 38 (2010)

Proceedings of the Fifth International Conference on Graph Transformation - Doctoral Symposium (ICGT-DS 2010)

Local Confluence Analysis of Consistent EMF Transformations

Enrico Biermann

16 pages

Guest Editor: Andrea Corradini
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Local Confluence Analysis of Consistent EMF Transformations

Enrico Biermann

Department of Software Engineering and Theoretical Computer Science
Technische Universität Berlin, Germany
enrico@cs.tu-berlin.de

Abstract: Model transformation is one of the key activities in model-driven software development. An increasingly popular technology to define modeling languages is provided by the Eclipse Modeling Framework (EMF). Several EMF model transformation approaches have been developed, focusing on different transformation aspects. For the analysis of model transformations, graph transformation techniques provide a formal basis and tool support. In this paper we aim to make use of those techniques by providing a formal foundation of consistent EMF transformations to analyze critical pairs between EMF transformation rules as well as extending the notion of local confluence to EMF transformation systems. The analysis is also demonstrated on a small example simulating the firing behavior of elementary Petri nets.

Keywords: EMF, graph transformation critical pairs, consistency

1 Introduction

Model-driven software development is considered as a promising paradigm in software engineering [Sch06]. Models are the central artifacts in model-driven developments. Hence, inspecting and modifying models to reduce their complexity and improve their readability, maintainability and extensibility (i.e. by performing *model refactoring* [MT04]) are important issues of model development. Thus Model transformation can be considered as one of the key activities in model-driven software development.

The ECLIPSE Modeling Framework (EMF) [EMF11] has evolved to a de facto standard technology to define models and modeling languages. EMF provides a modeling and code generation framework for ECLIPSE applications based on structured data models. The modeling approach is similar to that of MOF, actually EMF supports Essential MOF (EMOF) as part of the OMG MOF 2.0 specification [Obj08]. Several EMF model transformation approaches have been developed, focusing on different transformation aspects.

EMF transformations are defined as a special kind of attributed typed graph transformations using node type inheritance. Most properties of EMF models and model instances can easily be mapped to the domain of attributed typed graphs. On the type level EClasses are represented as graph nodes in an attributed type graph. EAttributes, which model attributes of EClasses are represented as node attribute edges. Associations between classes are modeled by EReferences in EMF. Their corresponding representation in an attributed type graph are graph edges.

However, EMF models have several structural properties that don't have a direct correspondence in the domain of graphs. The most prominent one being containment relations, i.e. compositions in UML, define an ownership relation between objects. Thereby, they induce a hier-

archical structure in model instantiations. In MOF and EMF, this structure is further used to implement a mapping to XML, known as XMI (XML Meta data Interchange) [Obj08]. Containment always implies a number of constraints for model instantiations that must be ensured at run-time, e.g. there must be no containment cycles and only a maximum of one container for each object. Furthermore, some associations between classes may be bidirectional, in EMF this is modeled by the opposite property of an EReference. If there are two EReferences between the same two EClasses in opposite direction, they can be used to model a bidirectional association, if the opposite property of both EReferences points to the other EReference. Another property of EReferences is ”unique”. Which means, that on the instance level, there may only be one association of a given type between the same two objects. We assume, that the ”unique” property is true for all our EReferences, meaning, that on the graph level, there are no parallel edges.

In this paper, we formalize consistent EMF transformations by providing conditions on EMF transformation rules which ensure that the consistency of an EMF model instance (i.e. the satisfaction of EMF properties) is preserved when a rule is applied.

The aim of this paper is to make use of graph transformation techniques to detect conflicts in EMF transformations. Even though individual rules lead to correct transformation steps, one such step might lead to an EMF model on which a second rule is no longer applicable resulting in a conflict. In graph transformation theory such conflicts are detectable by critical pair analysis. However, the additional structural properties of EMF model instances are not taken into account during that analysis. We have to consider those properties towards an extended notion of local confluence for EMF transformations. Based on this notion the rich theory of algebraic graph transformation can be applied to these EMF transformations to show functional behavior and correctness.

In Section 2 the notion of EMF graphs and consistent EMF transformations are introduced. Section 3 defines critical pairs for EMF transformations and presents the new result for confluence of EMF transformation systems. Later in Section 4 consistency and conflict analyses are demonstrated on an executional semantics for elementary Petri nets.

2 Formal foundation of EMF transformation by graph transformation

Conflict analysis for graph transformations has been investigated in [EEPT06][LEH+10]. However because of the additional structural constraints present in EMF models and instances it can not be directly used when using graph transformations to transform those models. The underlying graph representation of attributed typed graphs, described in Definition 1, has to be extended to model additional properties of EMF models and instances given in Definition 2. Please note, that even though it would be possible to define EMF transformations in a categorical setting, with objects being EMF graphs typed over a specific EMF type graph and morphisms being graph morphisms between those objects, we will not do so. In a category based on EMF graphs it would not be possible to construct pushouts in general because not every pushout object would be an EMF graph (e.g. it could have a containment cycle). Therefore, instead of defining EMF transformations as its own category, we use the existing category of attributed typed graphs as a starting base, distinguish certain subsets of the existing set-theoretic representation and define

conditions for those subsets that must be fulfilled. Since the base category is still attributed typed graphs we can take advantage of its theoretical results, if we can show the consistency for our properties.

Note that attributed type graphs are formalized as E-graphs in [EEPT06]. An E-graph is a special kind of graph, where aside from normal graph nodes and edges (denoted as TG_{V_G} and TG_{V_E}) there are edges that connect nodes and edges to data types. For example, a node with an attribute *name* is depicted as a normal graph node with a node attribute edge ($TG_{E_{NA}}$) called *name* to a data node (TG_{V_D}) called *String*, which is the only representative in the final DSIG-Algebra Z for the sort *string*.

Definition 1 (attributed type graph with inheritance) An attributed type graph with inheritance $ATGI = (TG, Z, I, A)$ consists of an attributed type graph $ATG = (TG, Z)$, where TG is an E-Graph $TG = (TG_{V_G}, TG_{V_D}, TG_{E_G}, TG_{E_{NA}}, TG_{E_{EA}}, (source_i, target_i)_{i \in \{G,NA,EA\}})$ with $TG_{V_D} = S'_D$ and final DSIG-Algebra Z;
an inheritance graph $I = (I_V, I_E, s, t)$ with $I_V = TG_{V_G}$; and a set $A \subseteq I_V$, called the abstract nodes.
For each node $n \in I_V$, the inheritance clan is defined by
$clan_i(n) = \{n' \in I_V | \exists \text{ path } n' \xrightarrow{*} n \text{ in } I\} \subseteq I_V$ with $n \in clan_I(n)$.
We write $n' \leq n$ for $n' \in clan_I(n)$ and say that n' *inherits from* n.
In the following definitions, we will use the shorter notion T_N for the set of type graph nodes TG_{V_G} and T_E for the set of type graph edges TG_{E_G}. Furthermore, $source_G$ and $target_G$ will be shortened to s and t.

An EMF type graph extends an attributed type graph by additional concepts, such as containment edges and opposite edges.

Definition 2 (EMF type graph) An EMF type graph $ETG = (ATGI, C, OE)$ consisting of an attributed type graph with inheritance $ATGI = (TG, Z, I, A)$, a set $C \subseteq T_E$ of *containment edges*, and a relation $OE \subseteq T_E \times T_E$ of opposite edges.

- We define a *containment relation* $contains_{TG} \subseteq T_N \times T_N$:[1]

  $contains_{TG} =$
  $\{(n,m) \mid \exists c \in C : n < s(c) \wedge\ m < t(c)\} \cup$
  $\{(x,y) \mid \exists z \in T_N : (x\ contains_{TG}\ z \wedge z\ contains_{TG}\ y)\}$

  Now we can define a relation containing cyclic containments only:

  $cycle_{TG} = \{(x,y) \in contains_{TG} | (y,x) \in contains_{TG}\}$

  Based on $cycle_{TG}$, a subset of containment edges, called *cycle-capable containment edges*, is defined whose instances might be part of containment cycles:

  $C_{Cycle} = \{c \in C \mid \exists n < s(c) \wedge \exists m < t(c) : (n,m) \in cycle_{TG}\}$.

[1] If there is no confusion, we use infix notation for $contains_{TG}$, e.g. $(x\ contains_{TG}\ y)$ instead of $(x,y) \in contains_{TG}$.

- The relation OE obeys the following axioms:
  (opposite directions): for all $(e_1,e_2) \in OE : s(e_1) = t(e_2) \wedge t(e_1) = s(e_2)$,
  (anti-reflexivity): for all $e \in T_E : (e,e) \notin OE$,
  (symmetry): for all $(e_1,e_2) \in OE : (e_2,e_1) \in OE$, and
  (functionality): for $(e_1,e_2),(e_1,e_3) \in OE : e_2 = e_3$.

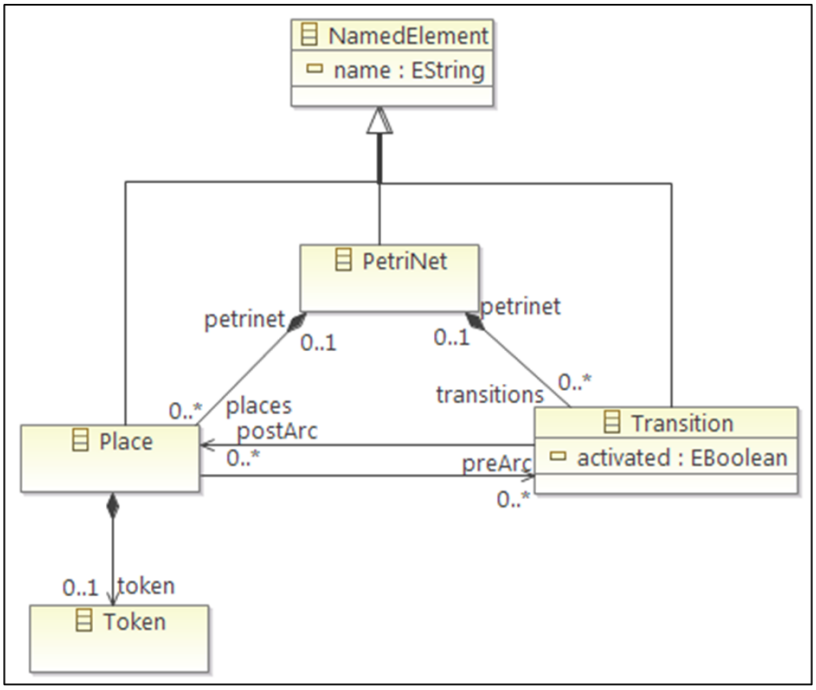


Figure 1: EMF model of elementary Petri nets as EMF type graph

Example 1 (EMF type graph) *In Figure 1 an example of an EMF type graph is shown. It describes elementary Petri nets, where each place may only have one token modeled as a containment relation. Places and transitions are both contained in the Petri net. Formally, the set of containment edges is given by* $C = \{places, transitions, token\}$ *and the relation* $OE = \{(petrinet, places), (places, petrinet), (transitions, petrinet'), (petrinet', transitions)\}$ *(note that* $petrinet'$ *is also named* $petrinet$ *in Figure 1 but is another entity with the same name). The relation* $contains_{TG}$ *would contain the pairs* $(PetriNet, Place), (Place, Token), (PetriNet, Token)$ *and* $(PetriNet, Transition)$*. Since the containment edges are strictly hierarchic (*PetriNet *contains* Place, Place *contains* Token*) the set for cycle-capable containment edges* C_{cycle} *is empty.*

Based on EMF type graphs we can define EMF graphs for EMF model instances which must fulfill certain properties being cycle-free on containment edges, having no parallel edges and valid opposite edges. Those properties are formally defined in Definition 3.

Definition 3 (EMF graph) Given an EMF type graph $ETG = (ATGI, C, OE)$, an attributed graph G and a morphism $type_G\colon G \to ATGI$.
Graph G is an *EMF graph* typed over ETG if the following conditions hold:

Let $G_C = \{e \mid type_G(e) \in C\}$ be the set of containment edges in G and

$contains_G = \{(s(e),t(e)) \mid \forall e \in G_C\} \cup$
$\{(x,z) \mid \exists y \in G_N : (x\ contains_G\ y) \wedge (y\ contains_G\ z)\}$ be the containment relation $G_N \times G_N$

(1) *(at most one container)*: $e_1, e_2 \in G_C : t_G(e_1) = t_G(e_2) \Rightarrow e_1 = e_2$,

(2) *(no containment cycles)*: $(x,x) \notin contains_G$ for all $x \in G_N$,

(3) *(all opposite edges)*: If $(e_1,e_2) \in OE$, then $\forall e \in G_E$ with $type_G(e) = e_1$ there is also an $e' \in G_E$ with $type_G(e') = e_2$ and $s_G(e) = t_G(e')$ and $s_G(e') = t_G(e)$,

(4) *(no parallel edges)*: For all $e_1, e_2 \in G_E$ with $s_G(e_1) = s_G(e_2)$, $t_G(e_1) = t_G(e_2)$, and $type_{G_E}(e_1) = type_{G_E}(e_2)$ we have $e_1 = e_2$.

An EMF graph G is called *rooted* if there is a node $r \in G_N$, called *root node* such that $\forall x \in G_N$ with $x \neq r : r\ contains_G\ x$.

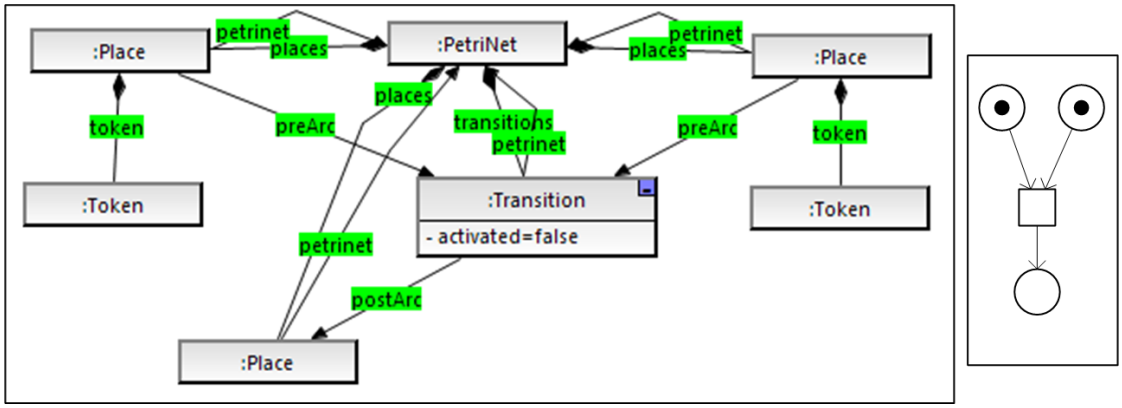


Figure 2: EMF model instance as an EMF graph

Example 2 (EMF graph) *Figure 2 shows an EMF graph for the EMF type graph given in Figure 1. All contained objects (all nodes except* PetriNet*) only have one container. There are no containment cycles and all edges that require an opposite (petrinet, places, transitions) have one. There are also no parallel edges of the same type between the same two nodes.*

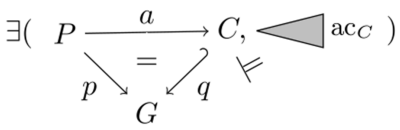
Transformation rules for EMF graphs must consist of EMF graphs themselves. In addition they must fulfill the properties given in Definition 6 to ensure that after each transformation step,

the EMF graph the rule was applied to is still a valid EMF graph. The application of a rule can be further restricted by an application condition AC, a logical formula over the left-hand side of a rule that must be fulfilled for the rule to be applicable. The theoretical results concerning application conditions have been discussed in [EHL+10]. Here we only give the definition for application conditions:

Definition 4 (application condition) An application condition ac_P over a graph P is inductively defined as follows:

- true is an application condition over P
- For every morphism $a : P \to C$ and every application condition ac_C over C, $\exists(a, ac_C)$ is an application condition over P
- For application conditions c, c_i over P with $i \in I$ (for all index sets I), $\neg c$ and $\bigwedge_{i \in I} c_i$ are application conditions over P.

We define inductively when a morphism satisfies an application condition: Every morphism satisfies true. A morphism $p : P \to G$ satisfies an application condition $\exists(a, ac_C)$, denoted $p \models \exists(a, ac_C)$, if there exists an injective morphism q such that $q \circ a = p$ and $q \models ac_C$. A morphism $p : P \to G$ satisfies $\neg c$ if p does not satisfy c and satisfies $\bigwedge_{i \in I} c_i$ if it satisfies each $c_i (i \in I)$.



Definition 5 (EMF transformation rule) A transformation rule typed over an EMF type graph $ETG = (ATGI, C, OE)$ is given by $p = (L \leftarrow K \rightarrow R, AC)$, where L, K and R are EMF graphs over ETG called left-hand side (L), intersection (K), and right-hand side (R) and AC is a condition over L. The morphisms $K \to L$ and $K \to R$ are injective. For better readability we will use inclusions to depict injective morphisms in the following definitions and use a subset notion for the rules $p = (L \supseteq K \subseteq R, AC)$.
We denote the sets of deleted / newly created elements, resp., by $L'_X := L_X - K_X$ and $R'_X := R_X - K_X$ (with $X = N, E$ or C).

Applying arbitrary transformation rules to EMF graphs may result in a graph, that violates one or more of the properties of an EMF graph given in Definition 3. Therefore, it is necessary to restrict the set of possible transformation rules to those that do preserve the properties of EMF graphs.
In addition, we want to forbid the creation or deletion of disconnected containment structures, because any rule, that can unconditionally create a containment connection between two nodes, could potentially violate property (1) and (2) of an EMF graph.

Definition 6 (Consistent transformation rule) Let $p = (L \supseteq K \subseteq R, AC)$ be a transformation rule as defined in Definition 5. Rule p is *consistent* wrt. containment and parallel edges if the

following constraints are satisfied:

(1) *(node deletion)*: $\forall n \in L'_N : \exists e \in L'_C$ with $t_L(e) = n$,

(2) *(node creation)*: $\forall n \in R'_N : \exists e \in R'_C$ with $t_R(e) = n$,

(3) *(containment edge deletion)*: $\forall e \in L'_C$ with $t_L(e) = n$:

$$n \in L'_N \quad \vee \quad (n \in K_N \wedge \exists e' \in R'_C \text{ with } t_R(e') = n)$$

(4) *(containment edge creation)*: $\forall e \in R'_C$ with $t_R(e) = n$:

$$n \in R'_N \quad \vee \quad (n \in K_N \wedge \exists e' \in L'_C \text{ with } t_L(e') = n)$$

(5) *(creation of cycle-capable containment edges)*:
$\forall (e : n \blacklozenge\!\!- m) \in R'_C$:

$$\exists (e' : o \blacklozenge\!\!- m) \in L'_C:$$
$$((o,n) \in contains_L \wedge (m,n) \notin contains_L) \vee (n,o) \in contains_L$$

(6) *(creation of parallel edges)*: $\forall (e : n \to m) \in R'_E - R'_C$ there is an $AC' = \neg\exists(L \to C', true)$ where C' contains an edge $(e' : n \to m)$ of the same type, i.e. $type_{C'_E}(e') = type_{R_E}(e)$ and there exists AC'' such that $AC = AC' \wedge AC''$

Condition (1) and (2) forbid the deletion or creation of nodes without simultaneously also deleting or creating a containment edge. For example a transformation rule for the Petri net model can not create a single *Place* that is not contained in a PetriNet. Condition (3) and (4) ensure, that for any deleted or created containment edge, either the content node was also deleted/created, or if the content node already existed, there was a containment edge in the lhs/rhs to that node, that was created/deleted. For example it is allowed to delete the containment edge *token* between a *Place* and a *Token*, if in the same rule, a new *token* edge is created between a new place and that *Token*. Condition (6) ensures that no parallel edges are created between the same two nodes. Note that this includes containment edges.
Condition (5) is the most complex condition and handles cycle-capable containment edges. Containment edges that can potentially introduce cycles, may only be shifted up or down within an existing containment hierarchy. Therefore it is necessary to completely specify the containment path in the rule, when ”moving” a cycle-capable containment edge. Figure 3 illustrates this situation. The dotted containment line between node o and n in this figure means that all nodes and containment edges between those two nodes must be present in the rule.

Example 3 (Consistent EMF transformation rule) *The rule shown in Figure 4 is a consistent EMF transformation rule. Condition (1) of Definition 6 is fulfilled because for the deleted node* Token, *there exists a deleted containment edge* token. *Condition (2), (4) and (5) are fulfilled because there are no created nodes, containment edges or cycle-capable containment edges in the rule. Condition (3) is fulfilled because for the deleted containment edge* token *the contained node* Token *is also deleted.*

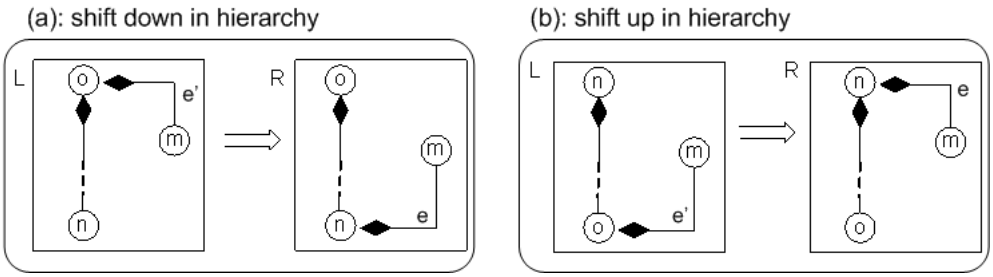


Figure 3: Creating cycle-capable containment edges

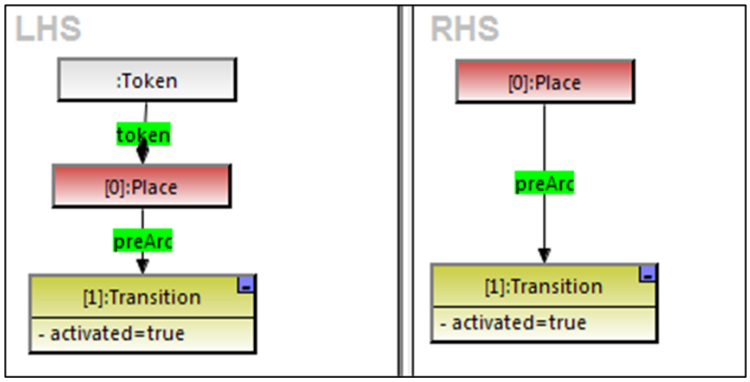


Figure 4: Rule RemovePre

The application of the rule will always yield a valid EMF graph as a result according to Theorem 1. The proof for this theorem is given in [BET08].

Theorem 1 (Consistent transformation step) *Given a consistent transformation rule* $p = (L \supseteq K \subseteq R, AC)$ *typed over* ETG *and a match* $L \xrightarrow{m} G$ *to an EMF graph* G *which is concretely typed by* $type_G\colon G \to TG$*. Then, the result graph* H *of the direct transformation* $G \overset{p,m}{\Longrightarrow} H$ *is an EMF graph.*

Given a transformation system with consistent EMF transformation rules, one of the interesting properties is whether the whole system is confluent or not, meaning whether different rule sequences lead to the same result or if there are conflicting situations where one rule prevents the application of another rule. Critical pair analysis for EMF transformations is discussed in the following section.

3 Local confluence analysis

Graph transformation theory allows the computation of conflicts between transformation rules with critical pair analysis. Critical pair analysis is known from term rewriting and can be used to check if a rewriting system contains conflicting computations. A conflict in graph transformation

theory is a situation, where two rules are applicable, but after one rule was applied the other is no longer applicable. Critical pairs formalize this idea, by showing a conflicting situation in a minimal context.
A set of consistent EMF transformation rules can be analyzed in a similar way. However, EMF transformations are not a distinct category but are defined over $Graph_{ATGI}$ category with additional properties.

Definition 7 Given consistent EMF transformation rules r_1 and r_2 and attributed typed graphs G_1, G_2 and K, then $G_1 \overset{r_1,o_1}{\Longleftarrow} K \overset{r_2,o_2}{\Longrightarrow} G_2$ is a critical pair for EMF graphs, if $G_1 \overset{r_1,o_1}{\Longleftarrow} K \overset{r_2,o_2}{\Longrightarrow} G_2$ is a critical pair in $Graph_{ATGI}$ and G_1, G_2 and K are EMF graphs.

Definition 7 defines critical pairs for EMF graphs. However it remains to be shown that these critical pairs are complete i.e. every possible conflict between two consistent EMF transformation rules embeds a critical pair according to Definition 7. To show the completeness for critical pairs for EMF graphs, two questions must be answered:

(1) Is there a critical pair in $Graph_{ATGI}$ where K is not an EMF graph that is embedded in a conflict between two EMF transformation rules?

(2) Do the additional constraints on EMF graphs cause new conflicts not reflected by the critical pairs for $Graph_{ATGI}$?

Lemma 1 addresses the question, whether the definition of EMF critical pairs in Definition 7 is complete i.e. whether every possible conflicts between two rules embed an EMF critical pair or if there are other conflicts not detectable by those critical pairs.

Lemma 1 (Completeness of critical pairs for EMF graphs) *Critical pairs for EMF graphs are complete in the sense that for every conflict $E_1 \overset{r_1,m_1}{\Longleftarrow} E \overset{r_2,m_2}{\Longrightarrow} E_2$ there is a critical pair for EMF graphs $G_1 \overset{r_1,o_1}{\Longleftarrow} K \overset{r_2,o_2}{\Longrightarrow} G_2$ according to Definition 7 that can be embedded in this conflict.*

Proof. We have to show, that

(1) If K is not EMF graph then $G_1 \overset{r_1,o_1}{\Longleftarrow} K \overset{r_2,o_2}{\Longrightarrow} G_2$ is not a critical pair for EMF graphs:
Since K is not an EMF graph, it must violate one of the conditions given in Definition 3.

- *(at most one container)*: A subgraph with a node that has more than one container can't be embedded into a valid EMF graph. Therefore it is never a conflicting situation for an EMF graph.
- *(no containment cycles)*: A subgraph with a containment cycle can't be embedded into a valid EMF graph. Therefore it is never a conflicting situation for an EMF graph.
- *(all opposite edges)*: A K missing an opposite edge could be embedded into a valid EMF graph. However K is an overlapping of two EMF graphs (L_1 of r_1 and L_2 of r_2). Therefore no overlapping K can contain singular opposite edges without an opposite.

– *(no parallel edges)*: A subgraph with two parallel edges can't be embedded into a valid EMF graph. Therefore it is never a conflicting situation for an EMF graph.

(2) There are no conflicts for EMF graphs without a corresponding critical pair in $Graph_{ATGI}$: Assuming there is a conflicting transformation between r_1 and r_2, the conflict reason must be one of the conditions for EMF Graphs given in Definition 3.

– *(at most one container)*: The potential conflict would be the creation of a container edge by both rules. However both rules are consistent EMF transformation rules and according to Cond. 4 of Definition 6 can't create containment edges without creating a contained node or deleting an existing containment edge as part of the rule. Therefore this conflict does not occur for consistent EMF transformation rules.

– *(no containment cycles)*: The potential conflict would be the introduction of a containment cycle. Both rules are consistent EMF transformation rules and K is an EMF graph. Therefore all derivations of K via r_1 and r_2 are EMF graphs which don't have containment cycles.

– *(all opposite edges)*: There is no possible conflict.

– *(no parallel edges)*: The potential conflict would be the creation of an edge which prevents the other rule from also creating an edge. This is also a conflict in $Graph_{ATGI}$ because according to Cond. 6 of Definition 6, there exists an AC which forbids an existing edge between the two nodes where the edge should be created.

□

Lemma 1 shows the completeness of critical pairs for EMF graphs. Together with Definition 7 we know that all critical pairs for EMF graphs are a subset of critical pairs of attributed typed graphs. Therefore we can now state the theorem concerning local confluence for EMF transformation systems. Local confluence is given if all critical pairs are are strictly confluent which means that no matter which rule has been applied in a conflicting situation, further rule applications will eventually lead to a common graph again.

Theorem 2 (Local confluence for EMF transformation systems) *An EMF transformation system is locally confluent if all its critical pairs are strictly confluent.*

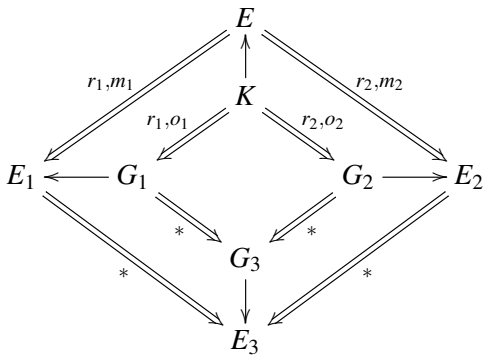
Proof. Given two EMF transformations $E_1 \overset{r_1,m_1}{\Longleftarrow} E \overset{r_2,m_2}{\Longrightarrow} E_2$ those two transformations are either independent or dependent(in conflict).

(1) independent: because of the Local-Church-Rosser theorem ([EEPT06]), there are transformations $E_2 \overset{r_1,m_1'}{\Longrightarrow} E_3$ and $E_1 \overset{r_2,m_2'}{\Longrightarrow} E_3$

(2) conflicting:

$$\begin{array}{ccccc} G_1 & \overset{r_1,o_1}{\Longleftarrow} & K & \overset{r_2,o_2}{\Longrightarrow} & G_2 \\ \downarrow & & \downarrow & & \downarrow \\ E_1 & \underset{r_1,m_1}{\Longleftarrow} & E & \underset{r_2,m_2}{\Longrightarrow} & E_2 \end{array}$$

Because of the completeness of critical pairs for EMF transformations (Lemma 1) there is a critical pair $G_1 \overset{r_1,o_1}{\Longleftarrow} K \overset{r_2,o_2}{\Longrightarrow} G_2$ that is embedded in this conflict.



Critical pairs are strictly confluent by assumption, therefore $\exists G_1 \overset{*}{\Longrightarrow} G_3$ and $G_2 \overset{*}{\Longrightarrow} G_3$.
Furthermore G_3 is an EMF graph according to Theorem 1. Because of the Embedding theorem ([EEPT06]) $\exists G_1 \overset{*}{\Longrightarrow} G_3$ and $G_1 \to E_1$ implies $E_1 \overset{*}{\Longrightarrow} E_3$ and $G_3 \to E_3$. Analogue for $E_2 \overset{*}{\Longrightarrow} E_3$.

□

4 Example: Petri net firing

To illustrate the results, we demonstrate the analysis on an interpreter semantics of elementary Petri nets. Elementary Petri nets may only have a single token on each place. We modeled the example using our EMF transformation tool HENSHIN [BESW10]. In Figure 1 our Petri net model is shown. Each place can have at most one token. Containments are used in an intuitive way to model places and transitions as part of the Petri net and token as part of the place. Transitions have an attribute *activated* which is used by the rules to mark transitions that could execute a firing step.

The semantics of elementary Petri nets is modeled by four transformation rules. Rule *ActivateTransition* in Figure 5 is called first and marks a transition as activated. It has two application conditions that ensure that all pre places contain a token and no post place does. The application condition ¬∃ *PostFull* corresponds to a simple negative application condition. However, ¬∃ *PreFree* has another nested condition ¬∃ *HasNoToken*. The rule matches any transition, where there is no post place with a token (PostFull) and there is no pre place (PreFree), that has no token (HasNoToken). [2]

ActivateTransition is followed by two rules that are applied as long as possible, *RemovePre* (see Figure 4) which removes token from pre and *AddPost* (see Figure 6) which adds a token to each post place of an activated transition.

Finally, rule *DeactivateTransition*, shown in Figure 7, is called which removes the activated status from a transition. The whole process of a firing step can be seen in concrete syntax in Figure 8. (1) shows the net before any rule is applied. (2) shows the net after the application *ActivateTransition*, (3) after applying *RemovePre* and *AddPost* as long as possible and finally (4) the resulting net after the firing step finishes with *DeactivateTransition*.

ActivateTransition and *DeactivateTransition* obviously fulfill the consistency conditions given in Definition 6. So does *RemovePre* as shown in Example 3. *AddPost* also fulfills the conditions similar to *RemovePre*.

For the actual critical pair analysis, we use the graph transformation and analysis tool AGG [AGG11]. Using the correspondence between EMF transformation and algebraic graph trans-

[2] Note that a transition with the same place in the pre domain and post domain is not enabled because the post domain is not empty. The rule reflects this behavior.

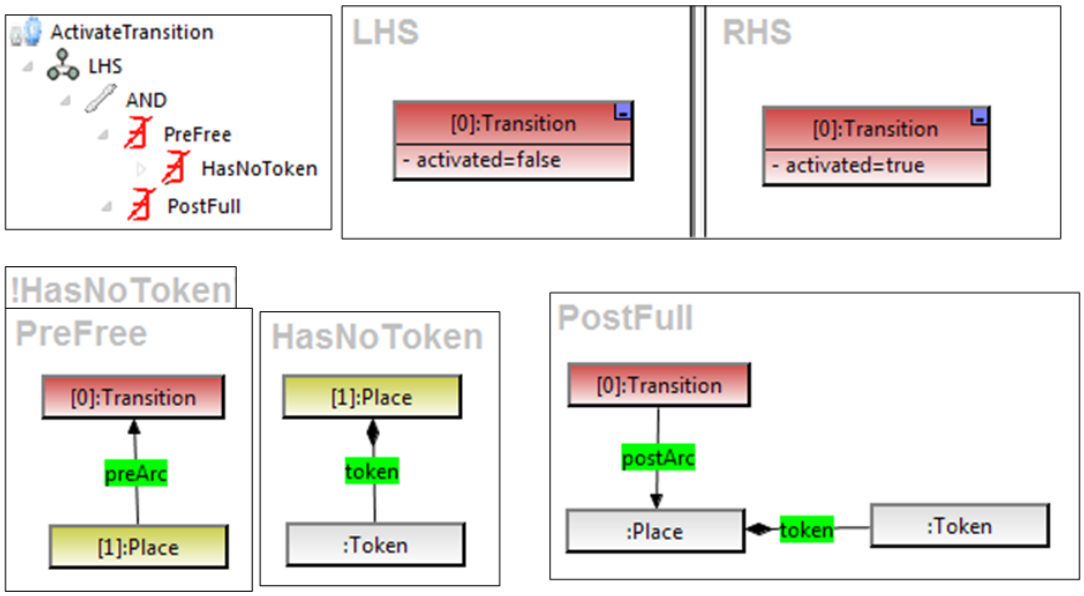


Figure 5: Rule ActivateTransition

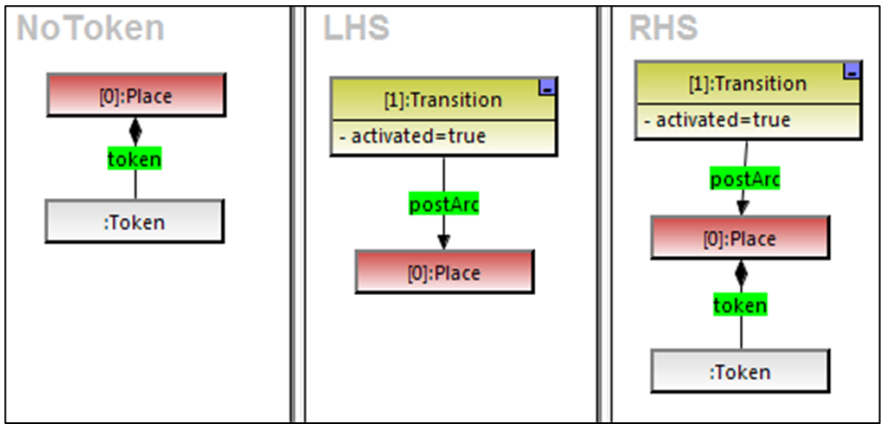


Figure 6: Rule AddPost

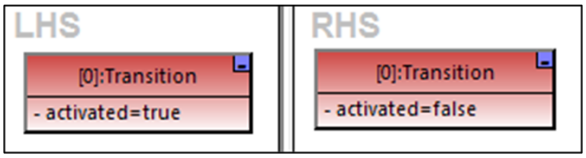


Figure 7: Rule DeactivateTransition

formations, the transformation rules are translated to AGG. In Figure 9, the rule *RemovePre* is shown in AGG. Since AGG does not have a notion for containment, the containment edge

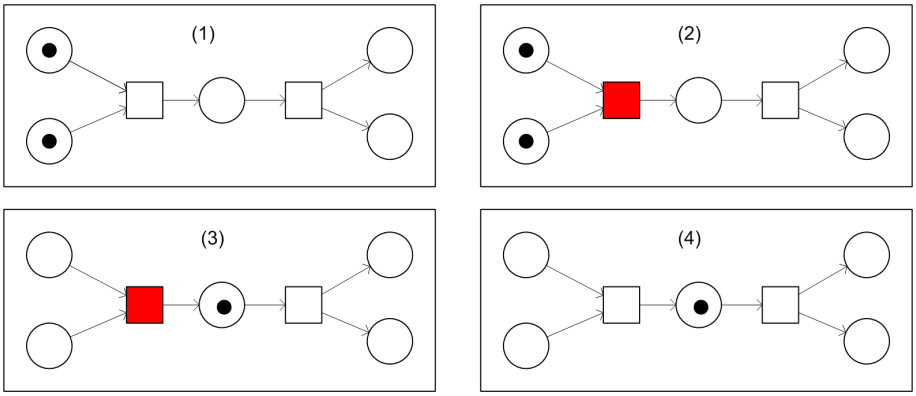


Figure 8: Firing step in concrete syntax

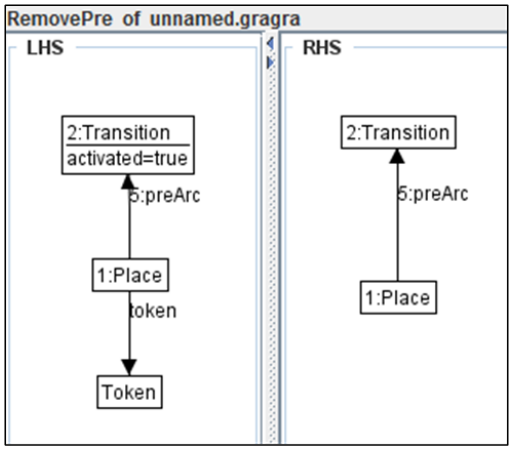


Figure 9: Rule RemovePre in AGG

token is a normal edge here. The rules themselves can now be analyzed for critical pairs by AGG. At the time of writing this paper, AGG could not analyze transformation rules with nested application conditions. Therefore it is only possible to compute the critical pairs for the rules *RemovePre*, *AddPost* and *DeactivateTransition* but not for *ActivateTransition*. The conflicts for the rule *RemovePre* can be seen in Figure 10. Critical pairs can be analyzed for the following properties:

- Whether the given conflict can occur at all, i.e. some conflicts depicted by a critical pair may never occur in syntactical correct instances which depends on the intention of the modeled language.
- if the critical pair is confluent, i.e. even though the direct rule applications lead to two different graphs, the system might still be confluent because a common graph can be achieved by multiple transformation steps.

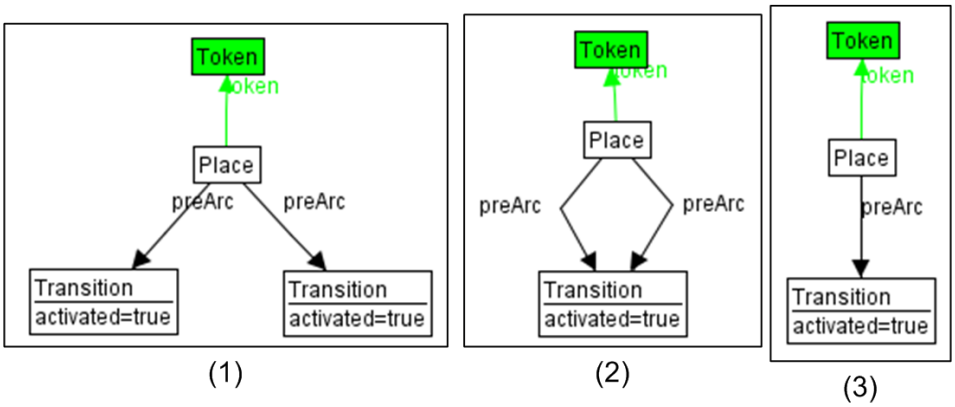


Figure 10: Critical pairs for rule pair RemovePre/RemovePre

The critical pair (1) denotes a situation where a place is in the pre domain of two activated transitions. In a simulation scenario this is a valid concern since such a situation would result in an invalid firing step for at least one of the transitions. To prevent this it is useful to restrict the application of *ActivateTransition* to situations where there is not yet a transition activated. This could be achieved by adding an application condition to the rule *ActivateTransition* that checks whether there is already an activated transition. However, this critical pair is more a semantical concern about firing steps in general. From a purely syntactical perspective, the critical pair is strictly confluent, because whether the left transition consumes the token or the right one, the resulting graphs when applying *RemovePre* to the left or right transition are isomorphic and can be joined by applying an empty rule sequence to either one.
Critical pair (2) denotes a situation where a place is connected with two different *preArcs* to the same transition. This situation is not a valid EMF graph according to Definition 3 and therefore can't occur in an EMF model and can be ignored. Critical pair (3) is a complete overlapping of *RemovePre* with itself and strictly confluent. In general, if the complete overlapping of a rule with itself is a critical pair, it is still confluent because whether you apply copy one of the rule or copy two, the resulting graphs are isomorphic and can be joined with an empty rule sequence.

5 Related work

There are several approaches and tools to define and execute transformation of EMF models and model instances such as ATL [JK05], EWL [KPPR07], Tefkat [LS05], VIATRA2 [VB07], MOMEMT [Bor07]. Most of these tools use a textual language to describe model changes with its own execution semantics. Others like VIATRA2 use graph transformation but neither do they have a notion for consistency nor further analysis techiques based on such a set of consistent transformation rules.

The given consistency constraints for EMF transformation rules ensure the validity of EMF model instances during transformations. Although this limits the expressiveness of transformation rules compared to other approaches, most of the removed transformations are not desired anyway because they violate EMF properties. An interesting case of transformations is the dis-

connection of subtrees which can not be expressed by a single consistent transformation rule. Instead each object in the subtree must be deleted explicitly before finally deleting the containment edge with the subtree root object.

In contrast to the model transformation approaches listed above, MOMENT as well as our approach have a formal basis. MOMENT is based on Maude which might be exploited for validation of EMF model transformations. All EMF model transformation tools including HENSHIN allow for general transformations because they support arbitrary changes to the containment structure. However to the best of our knowledge, apart from HENSHIN, none of the transformation approaches support the checking of consistency conditions for transformation rules as presented in this paper. Therefore, other transformation approaches are not well suited for formal analyses such as local confluence and termination.

6 Conclusion

In this paper we have shown the definition of consistent EMF transformation rules. By restricting the creation and deletion of containment edges we were able to create a subset of transformation rules that preserve the hierarchical structure of EMF model instances induced by containment edges. Furthermore, we defined critical pairs for EMF graphs and showed their completeness. With this result we were able to apply the local confluence results of graph transformations to EMF transformations.

Future work includes filtering the resulting critical pairs from AGG not by hand, but automatically by graph constraints. Furthermore we want to extend these results to transformation units [BESW10] which define a controlled application of transformation rules. Given proper termination criteria, the local confluence analysis supports the verification of the confluence of these transformation units. Concerning our EMF transformation tool HENSHIN, we plan to import the analysis results from AGG into HENSHIN in order to visualize them as conflict and dependency graphs on EMF transformation rules.

References

[AGG11] TFS-Group, TU Berlin. AGG. 2011. http://tfs.cs.tu-berlin.de/agg.

[BESW10] E. Biermann, C. Ermel, J. Schmidt, A. Warning. Visual Modeling of Controlled EMF Model Transformation using Henshin. *ECEASST* 32:1–14, 2010. http://journal.ub.tu-berlin.de/eceasst/issue/view/43

[BET08] E. Biermann, C. Ermel, G. Taentzer. Precise Semantics of EMF Model Transformations by Graph Transformation. In Czarnecki (ed.), *Proc. ACM/IEEE 11th International Conference on Model Driven Engineering Languages and Systems (MoDELS'08)*. LNCS 5301, pp. 53–67. Springer, 2008. http://tfs.cs.tu-berlin.de/publikationen/Papers08/BET08.pdf

[Bor07] A. Boronat. *MOMENT: A Formal Framework for Model Management*. PhD thesis, Universitat Politècnica de València, 2007.

[EEPT06] H. Ehrig, K. Ehrig, U. Prange, G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. EATCS Monographs in Theor. Comp. Science. Springer, 2006. http://www.springer.com/3-540-31187-4

[EHL+10] H. Ehrig, A. Habel, L. Lambers, F. Orejas, U. Golas. Local Confluence for Rules with Nested Application Conditions. In Ehrig et al. (eds.), *Proceedings of Intern. Conf. on Graph Transformation (ICGT' 10)*. LNCS 6372, pp. 330–345. Springer, 2010. http://www.springerlink.com/index/X273147851566804.pdf

[EMF11] Eclipse Consortium. Eclipse Modeling Framework (EMF) – Version 2.7. 2011. http://www.eclipse.org/emf.

[JK05] F. Jouault, I. Kurtev. Transforming Models with ATL. In *MoDELS Satellite Events*. LNCS 3844, pp. 128–138. Springer, Berlin, 2005.

[KPPR07] D. Kolovos, R. Paige, F. Polack, L. Rose. Update Transformations in the Small with Epsilon Wizard Language. *Journal of Object Technology* 6(9):53–69, 2007. http://www.jot.fm/issues/issues_2007_9/paper3

[LEH+10] L. Lambers, H. Ehrig, A. Habel, F. Orejas, U. Golas. Local Confluence for Rules with Nested Application Conditions based on a New Critical Pair Notion. Technical report 2010/07, Technical University of Berlin, 2010. http://www.eecs.tu-berlin.de/fileadmin/f4/TechReports/2010/tr_2010-07.pdf

[LS05] M. Lawley, J. Steel. Practical Declarative Model Transformation with Tefkat. In *MoDELS Satellite Events*. LNCS 3844, pp. 139–150. Springer, Berlin, 2005.

[MT04] T. Mens, T. Tourwé. A Survey of Software Refactoring. *Transactions on Software Engineering* 30(2):126–139, 2004.

[Obj08] Object Management Group. Meta Object Facility (MOF) Core Specification Version 2.0. http://www.omg.org/technology/documents/modeling_spec_catalog.htm#MOF, 2008.

[Sch06] D. C. Schmidt. Model-Driven Engineering. *IEEE Computer* 39(2):25–31, 2006.

[VB07] D. Varró, A. Balogh. The model transformation language of the VIATRA2 framework. *Science of Computer Programming* 68(3):214–234, 2007. doi:http://dx.doi.org/10.1016/j.scico.2007.05.004